7)

Aim:

Basic Spring Boot Application with Spring Data JPA

Description:

In this experiment, we will create a Spring Boot application that connects to a MySQL database and uses Spring Data JPA to perform basic database operations. The application will allow inserting and retrieving student records through a RESTful API.

- **Student.java** – The entity class representing students.
- **StudentRepository.java** – The JPA repository interface for database operations.
- **StudentController.java** – REST controller for handling HTTP requests.
- **StudentApplication.java** – Main application class for bootstrapping the application.
- **application.properties** – Configuration file for database and server.
- **pom.xml** – Maven configuration file for dependencies.

Program:

StudentApplication.java

```
package com.example;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
@SpringBootApplication
public class StudentApplication {
        public static void main(String[] args) {
                SpringApplication.run(StudentApplication.class, args);
        }
        @Bean
   CommandLineRunner initDatabase(StudentRepository repo) {
      return args -> {
        repo.save(new Student(1, "Rakesh kumar"));
        repo.save(new Student(2, "Murali"));
        repo.save(new Student(3, "vamsi"));
        System.out.println("Students inserted!");
      };
        }
}
```

Student.java

```
package com.example;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
@Entity
public class Student {
        @Id
   private int sno;
   private String sname;

   public Student() {}
   public Student(int sno, String sname) {
      this.sno = sno;
      this.sname = sname;
   }

   public int getSno() { return sno; }
   public void setSno(int sno) { this.sno = sno; }
   public String getSname() { return sname; }
   public void setSname(String sname) { this.sname = sname; }

}
```

application.properties

```
spring.application.name=Student
server.port= 9640
spring.datasource.url=jdbc:mysql://localhost:3306/mca
spring.datasource.username=root
spring.datasource.password= Pradeep@79979
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
```

StudentController.java

```
package com.example;
import java.util.List;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
//@RequestMapping("/students")
public class StudentController {
        private final StudentRepository repo;

   public StudentController(StudentRepository repo) {
      this.repo = repo;
   }
   //@RequestMapping("/students")
   // Add new student
   @PostMapping
   public Student addStudent(@RequestBody Student student) {
      return repo.save(student);
   }

   // Get all students
   @GetMapping
   public List<Student> getAllStudents() {
      return repo.findAll();
   }

}
```

StudentRepository.java (Interface)

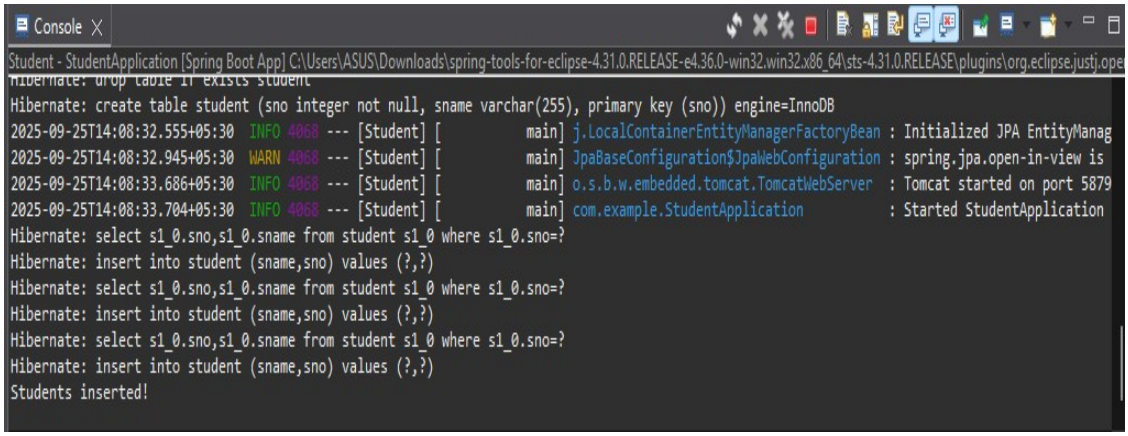
```
package com.example;
import org.springframework.data.jpa.repository.JpaRepository;
public interface StudentRepository extends JpaRepository<Student, Integer>{
}
```

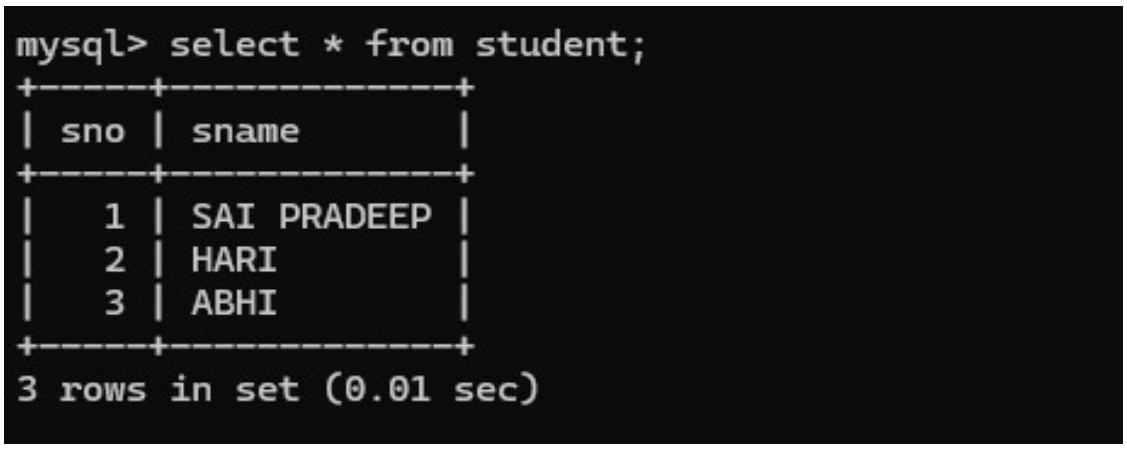
pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>3.5.4</version>
		<relativePath/> <!-- lookup parent from repository -->

	</parent>
	<groupId>com</groupId>
	<artifactId>StudentApplication</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>Student</name>
	<description>Demo project for Spring Boot</description>
	<url/>

	<licenses>
		<license/>
	</licenses>
	<developers>
		<developer/>
	</developers>

	<scm>
		<connection/>
		<developerConnection/>
		<tag/>
		<url/>
	</scm>

	<properties>
		<java.version>21</java.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jdbc</artifactId>
```

```
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		<dependency>
			<groupId>com.mysql</groupId>
			<artifactId>mysql-connector-j</artifactId>
			<scope>runtime</scope>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
		<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-
starter-data-jpa -->
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-data-jpa</artifactId>
   <version>3.5.2</version>
</dependency>

	</dependencies>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

</project>
```

Output:


Console
Student - StudentApplication [Spring Boot App] C:\Users\ASUS\Downloads\spring-tools-for-eclipse-4.31.0.RELEASE-e4.36.0-win32.win32.x86_64\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.ope
Hibernate: drop table if exists student
Hibernate: create table student (sno integer not null, sname varchar(255), primary key (sno)) engine=InnoDB
2025-09-25T14:08:32.555+05:30 INFO 4068 --- [Student] [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManag
2025-09-25T14:08:32.945+05:30 WARN 4068 --- [Student] [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is
2025-09-25T14:08:33.686+05:30 INFO 4068 --- [Student] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 5879
2025-09-25T14:08:33.704+05:30 INFO 4068 --- [Student] [main] com.example.StudentApplication : Started StudentApplication
Hibernate: select s1_0.sno,s1_0.sname from student s1_0 where s1_0.sno=?
Hibernate: insert into student (sname,sno) values (?,?)
Hibernate: select s1_0.sno,s1_0.sname from student s1_0 where s1_0.sno=?
Hibernate: insert into student (sname,sno) values (?,?)
Hibernate: select s1_0.sno,s1_0.sname from student s1_0 where s1_0.sno=?
Hibernate: insert into student (sname,sno) values (?,?)
Students inserted!


mysql> select * from student;
+-----+-------------+
| sno | sname |
+-----+-------------+
1	SAI PRADEEP
2	HARI
3	ABHI
+-----+-------------+
3 rows in set (0.01 sec)

8)

Aim:

Pagination and Sorting in Spring Data JPA

Description:

In this experiment, we will create a Spring Boot application that demonstrates how to paginate and sort database records using Spring Data JPA. We will use a Book entity with sample data, a JPA repository interface for database operations, and a REST controller to handle requests. Pagination parameters (page, size) and sorting parameters (sortBy, direction) will be passed via URL query parameters to retrieve data in a paginated and sorted manner.

Program:

application.properties

```
spring.application.name=Book
spring.datasource.url=jdbc:mysql://localhost:3306/new
spring.datasource.username=root
spring.datasource.password=Pradeep@79979
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
server.port=8844
```

BookApplication.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class BookApplication {

	public static void main(String[] args) {
		SpringApplication.run(BookApplication.class, args);
	}

}
```

Book.java

```
package com.example.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Book {
        @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String title;
    private String author;

    public Book() {}

    public Book(String title, String author) {
        this.title = title;
        this.author = author;
    }

    @Override
    public String toString() {
        return "Book{id=" + id + ", title='" + title + '', author='" + author + '}';
    }

    // getters and setters
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }

    public String getAuthor() { return author; }
    public void setAuthor(String author) { this.author = author; }

}
```

BookRepository (Interface)

```
package com.example.demo;
import org.springframework.data.jpa.repository.JpaRepository;
public interface BookRepository extends JpaRepository<Book, Long>{
}
```

BookController.java

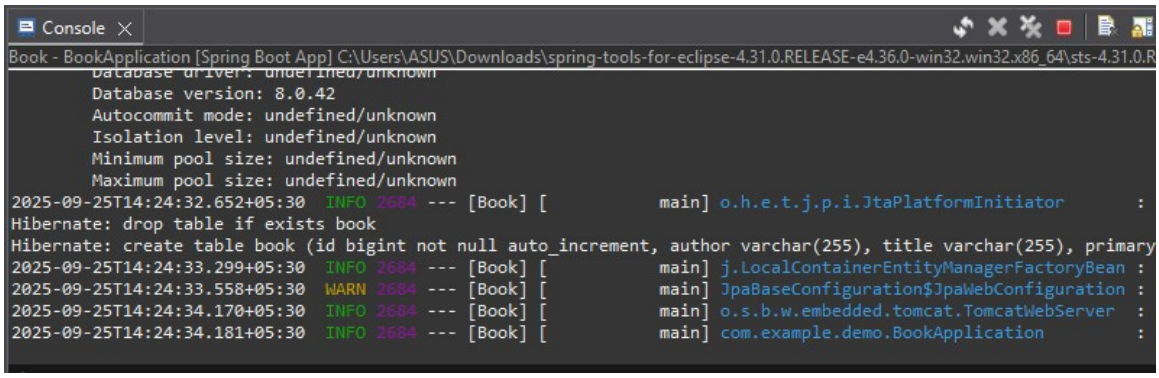
```
package com.example.demo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.web.bind.annotation.*;
@RestController
@RequestMapping("/books")
public class BookController {
        @Autowired
   private BookRepository bookRepository;
   @GetMapping("/init")
   public String initData() {
      if (bookRepository.count() == 0) {
         bookRepository.save(new Book("Spring Boot Basics", "John"));
         bookRepository.save(new Book("Java Programming", "Alice"));
         bookRepository.save(new Book("Hibernate in Action","Bob"));
         bookRepository.save(new Book("Microservices Guide", "Carol"));
         bookRepository.save(new Book("Data Structures", "Davidraj"));}
      return "Sample books added!";
   }
   @GetMapping
   public Page<Book> getBooks(
         @RequestParam(defaultValue = "0") int page,
         @RequestParam(defaultValue = "3") int size,
         @RequestParam(defaultValue = "title") String sortBy,
         @RequestParam(defaultValue = "asc") String direction
   ) {
      Sort sort = direction.equalsIgnoreCase("asc") ?
            Sort.by(sortBy).ascending() :
            Sort.by(sortBy).descending();
      Pageable pageable = PageRequest.of(page, size, sort);
      return bookRepository.findAll(pageable);
   }
}
```

pom.xml

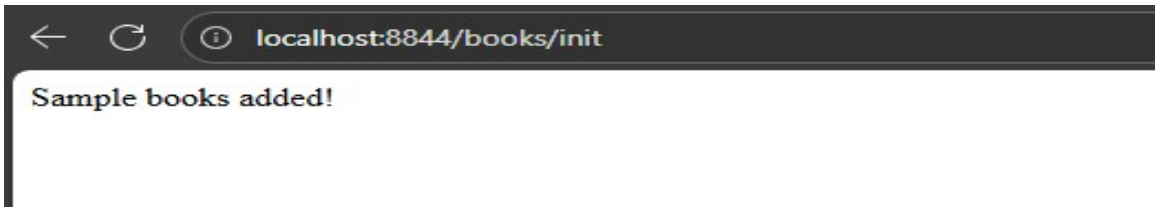
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.5.6</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <groupId>com</groupId>
    <artifactId>PaginationandSortingApplication</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>Book</name>
    <description>Demo project for Spring Boot</description>

    <licenses>
        <license/>
    </licenses>

    <developers>
        <developer/>
    </developers>

    <scm>
        <connection/>
        <developerConnection/>
        <tag/>
        <url/>
    </scm>

    <properties>
        <java.version>21</java.version>
    </properties>
```

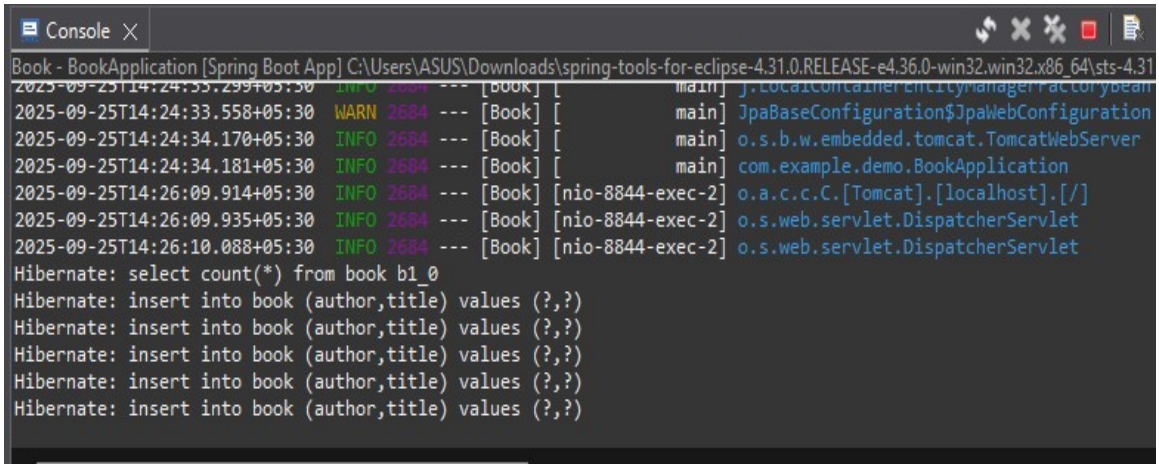
```
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jdbc</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>com.mysql</groupId>
            <artifactId>mysql-connector-j</artifactId>
            <scope>runtime</scope>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Output:

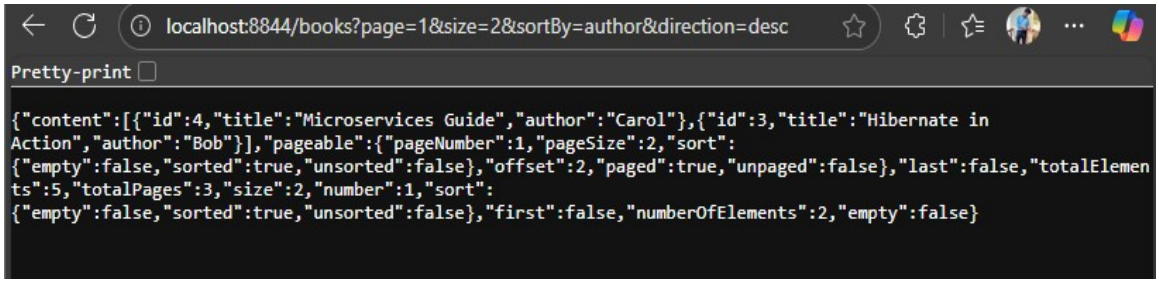


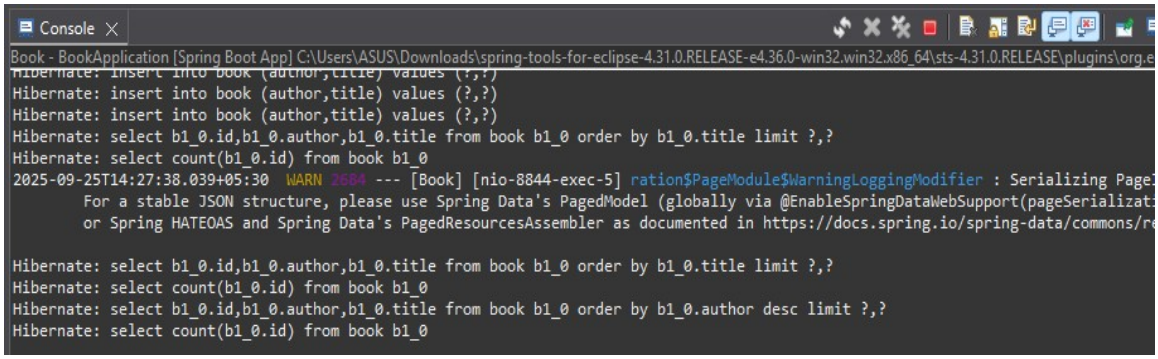

```
mysql> use new;
Database changed
mysql> show tables;
+---------------+
| Tables_in_new |
+---------------+
| book          |
+---------------+
1 row in set (0.10 sec)

mysql> select * from book;
Empty set (0.06 sec)
```





```
mysql> use new;
Database changed
mysql> select * from book;
Empty set (0.94 sec)

mysql> select * from book;
+----+----------+--------------------+
| id | author   | title              |
+----+----------+--------------------+
|  1 | John     | Spring Boot Basics |
|  2 | Alice    | Java Programming   |
|  3 | Bob      | Hibernate in Action |
|  4 | Carol    | Microservices Guide |
|  5 | Davidraj | Data Structures    |
+----+----------+--------------------+
5 rows in set (0.07 sec)
```





9)

Aim:

Implementing AOP for Logging with Spring Data JPA

Description:

In this experiment, we create a Spring Boot application to manage products. The application includes:

- **Entity** – Product with id, name, and price.
- **Repository** – ProductRepository for database operations.
- **Service** – ProductService to handle business logic.
- **Controller** – ProductController for REST APIs.
- **Aspect** – LoggingAspect to log method calls in ProductService.
- **Database** – H2 in-memory DB or MySQL.
- **Dependency Management** – Managed via Maven (pom.xml).

This demonstrates the use of **Spring Data JPA**, **Spring AOP**, and **RESTful API development**.

Program:

ProductRepository.java (Interface)

```
package com.example.demo;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ProductRepository extends JpaRepository<Product, Long> {

}
```

ProductService.java

```
package com.example.demo;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class ProductService {
    private final ProductRepository repo;
    public ProductService(ProductRepository repo) {
        this.repo = repo;
    }
    public Product saveProduct(Product product) {
        return repo.save(product);
    }
    public List<Product> getAllProducts() {
        return repo.findAll();
    }
}
```

application.properties

```
spring.application.name=product
spring.datasource.url=jdbc:mysql://localhost:3306/mca
spring.datasource.username=root
spring.datasource.password=Pradeep@79979
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
server.port=8889
```

Product.java

```
package com.example.demo;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    private String name;
    private double price;
    public Product() {}
    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }
    // getters & setters
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public double getPrice() { return price; }
    public void setPrice(double price) { this.price = price; }
}
```

ProductController.java

```
package com.example.demo;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
@RequestMapping("/products")
public class ProductController {
    private final ProductService service;
    public ProductController(ProductService service) {
        this.service = service;
    }
    @PostMapping("/add")
    public Product addProduct(@RequestBody Product product) {
        return service.saveProduct(product);
    }
    @GetMapping("/all")
    public List<Product> getAllProducts() {
        return service.getAllProducts();
    }
}
```

LoggingAspect.java

```
package com.example.demo;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.springframework.stereotype.Component;


@Aspect

@Component

public class LoggingAspect {

   // Logs before executing any ProductService method

   @Before("execution(* com.example.demo.ProductService.*(..))")

   public void logBefore(JoinPoint joinPoint) {

      System.out.println(">>> Entering method: " + joinPoint.getSignature().getName());

   }

}
```

main.java

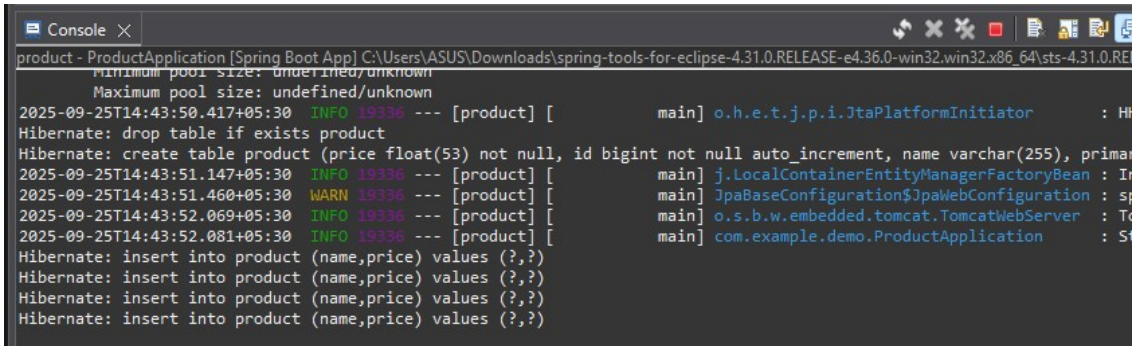
```
package com.example.demo;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
@SpringBootApplication
public class ProductApplication {
	public static void main(String[] args) {
		SpringApplication.run(ProductApplication.class, args);
	}
	 @Bean
	   CommandLineRunner runner(ProductRepository repo) {
	      return args -> {
	         repo.save(new Product("Laptop", 55000));
	         repo.save(new Product("Mobile", 20000));
	         repo.save(new Product("Tablet", 30000));
	         repo.save(new Product("Mouse", 35000));
	      };
	   }
}
```

pom.xml

```
≤?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>3.5.4</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com</groupId>
	<artifactId>productApplication</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>product</name>
	<description>Demo project for Spring Boot</description>
	<url/>
	<licenses>
		<license/>
	</licenses>
	<developers>
		<developer/>
	</developers>
	<scm>
		<connection/>
		<developerConnection/>
		<tag/>
		<url/>
	</scm>
	<properties>
		<java.version>21</java.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jdbc</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
```

```
                </dependency>
                <dependency>
                        <groupId>com.mysql</groupId>
                        <artifactId>mysql-connector-j</artifactId>
                        <scope>runtime</scope>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-test</artifactId>
                        <scope>test</scope>
                </dependency>
                    <!-- Spring AOP -->
        <dependency>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-starter-aop</artifactId>
        </dependency>
       <!-- Lombok (optional, just to reduce boilerplate) -->
        <dependency>
          <groupId>org.projectlombok</groupId>
          <artifactId>lombok</artifactId>
          <optional>true</optional>
        </dependency>
      <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
      </dependency>
           <!-- H2 Database (in-memory, no need for MySQL setup) -->
        <dependency>
          <groupId>com.h2database</groupId>
          <artifactId>h2</artifactId>
          <scope>runtime</scope>
        </dependency>
          </dependencies>
          <build>
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
          </build>
</project>
```

Output:




```
mysql> use mca;
Database changed
mysql> show tables;
+----------------+
| Tables_in_mca  |
+----------------+
| product        |
| student        |
+----------------+
2 rows in set (0.15 sec)

mysql> select * from product;
+--------+----+---------+
| price  | id | name    |
+--------+----+---------+
| 55000  |  1 | Laptop  |
| 20000  |  2 | Mobile  |
| 30000  |  3 | Tablet  |
| 35000  |  4 | Mouse   |
+--------+----+---------+
4 rows in set (0.03 sec)
```